Random Numbers Solutions

Copyright © James Raynard 2021

rand()

- Briefly describe some disadvantages of the traditional approach to generating random numbers
  - Some implementations provide poor quality random numbers
  - We often need to rescale the range (e.g. to obtain random numbers between 0 and 1)
  - This introduces “bias” in which some numbers appear more often than others
  - Also creates opportunities for programming errors(!)
  - Not suitable for secure applications as the sequence repeats too frequently
- Write a program which uses rand() to print out
  - A random floating-point number between 0 and 1
  - 10 random integers between 1 and 100

Copyright © James Raynard 2021

Random Numbers

- Give an outline description of how to obtain random numbers in modern C++
  - Create a random number engine instance
  - Optionally, this can be seeded using a random_device instance, or some other number which is not easily guessable
  - Create a suitable distribution instance, giving the range of numbers required as arguments
  - Call the () operator of the distribution, passing the engine instance as argument

Copyright © James Raynard 2021

Random number engine

- What is the purpose of a random number engine?
  - To generate a sequence of random numbers
  - Each time its () operator is called, the next number in the sequence is returned
- How do we use a random number engine?
  - A random number engine is implemented as a functor
  - To get a random number, we call the overloaded () operator of the engine
  - Each time we call it, we get the next number from the sequence
- Which random number engine is generally the best one to use?
  - mt19937 (Mersenne Twister)

Copyright © James Raynard 2021

Distribution

- What is the purpose of a distribution type?
  - Given a sequence of numbers, it will convert them so that they are within a given range and the probability of getting a certain number follows a given statistical distribution
- Which distributions are most useful when generating random numbers?
  - uniform_int_distribution and uniform_real_distribution
- Why are these particular distributions useful?
  - A uniform distribution produces a sequence in which each number in the given range is equally likely

Copyright © James Raynard 2021

Random number example

- Write a simple program which
  - Prints out ten integers with random values between 0 and 100
  - Prints out ten floating point numbers with random values between 0 and 1

Copyright © James Raynard 2021

std::random_device

- Explain how random_device generates random numbers
  - random_device uses information provided by the operating system (hardware device or system entropy)
- Why is random_device not suitable for generating large quantities of numbers?
  - If the operating system runs out of data, random_device has to wait until more data is available
- What is the most common use for random_device?
  - For seeding random_engine

Copyright © James Raynard 2021

Random engine usage

- Why is it good practice to make engine and distribution instances static or global?
  - Creating them has high overhead
  - Usually only one instance is needed per program
  - Creating a new engine instance will reset the sequence of random numbers, starting again from the first number. This may not produce the expected results

Copyright © James Raynard 2021